Managing Technology Risk in R&D Project Planning: Optimal Timing and Parallelization of R&D Activities

Pascale Crama
London Business School
Regent's Park, London NW1 4SA, United Kingdom
Tel. +44 20 7262 5050; Fax. +44 20 7724 7875; e-mail: pcrama@london.edu

Bert De Reyck
London Business School
Regent's Park, London NW1 4SA, United Kingdom
Tel. +44 20 7706 6884; Fax. +44 20 7724 7875; e-mail: bdereyck@london.edu

Zeger Degraeve
London Business School
Regent's Park, London NW1 4SA, United Kingdom
Tel. +44 20 7262 5050; Fax. +44 20 7724 7875; e-mail: zdegraeve@london.edu

Roel Leus
Katholieke Universiteit Leuven
Naamsestraat 69, 3000 Leuven, Belgium
Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium)
Tel. +32 16 32 69 67; Fax. +32 16 32 67 32; e-mail: Roel.Leus@econ.kuleuven.ac.be

— January 2005 —

Managing Technology Risk in R&D Project Planning: Optimal Timing and Parallelization of R&D Activities

Pascale Crama • Bert De Reyck • Zeger Degraeve • Roel Leus

Abstract

An inherent characteristic of R&D projects is technological uncertainty, which may result in project failure, and time and resources spent without any tangible return. In pharmaceutical projects, for instance, stringent scientific procedures have to be followed to ensure patient safety and drug efficacy in pre-clinical and clinical tests before a medicine can be approved for production. A project consists of several stages, and may have to be terminated in any of these stages, with typically a low likelihood of success. In project planning and scheduling, this technological uncertainty has typically been ignored, and project plans are developed only for scenarios in which the project succeeds. In this paper, we examine how to schedule projects in order to maximize their expected net present value, when the project activities have a probability of failure, and where an activity's failure leads to overall project termination. We formulate the problem, show that it is *NP*-hard and develop a branch-and-bound algorithm that allows to obtain optimal solutions. We also present polynomial-time algorithms for special cases, and present a number of managerial insights for R&D project and planning, including the advantages and disadvantages of parallelization of R&D activities in different settings.

Keywords

Project Management, project scheduling, risk

Industries, pharmaceutical, R&D projects

Analysis of algorithms, computational complexity, exact algorithms

Programming, integer, branch-and-bound, applications

1. Introduction

An important feature of Research and Development (R&D) projects is that, apart from the commercial and market risks common to all projects, their constituent activities also carry the risk of technical failure. Therefore, besides projects overrunning their budgets or deadlines and the commercial returns not meeting their targets, R&D projects also carry the risk of failing altogether, resulting in time and resources spent without any tangible return. In this paper, we tackle the problem of scheduling the activities of an R&D project that is subject to technological uncertainty, i.e. in which the individual activities carry a risk of failure, and where an activity's failure results in the project's overall failure. The goal is to schedule the activities in such a way as to maximize the expected net present value of the project, taking into account the activity costs, the cash flows generated by a successful project, the activity durations and the probability of failure of each of the activities.

The algorithms developed in this paper are useful in any R&D setting where activities carry a risk of failure, and are of particular interest to drug development projects in the pharmaceutical industry, in which stringent scientific procedures have to be followed to ensure patient safety in distinct stages, including pre-clinical and clinical tests, before a medicine can be approved for production. The project may need to be terminated in any of these stages, either because the product is revealed not to have the desired properties, or because of harmful side effects. The failure of one of the stages results in overall project termination. As stated by Gassmann et al. (2004), "If a drug candidate fails during the development phase it is withdrawn entirely from further testing. Unlike in the automobile industry, drugs are not modular products where a faulty stick shift can be replaced without throwing the entire car design away. In pharmaceutical R&D, drug design cannot be changed."

The contributions of this paper are the following. First, we introduce and formulate a generic model for optimally scheduling R&D project activities subject to technological uncertainty and technological dependencies, referred to as the *R&D Project Scheduling Problem* (*RDPSP*). Second, we show that the *RDPSP* is *NP*-hard, but that several special cases can be solved in polynomial time. Third, we develop a branch-and-bound algorithm that is capable of solving the *RDPSP* to optimality, we present computational tests demonstrating the capabilities of the algorithm and we discuss how the model and algorithms can be extended to take into account the risk preferences of the decision maker. Finally, we present a number of managerial insights based on the properties of the optimal solution, uncovering the benefits and disadvantages of parallelization of R&D activities in different settings.

2. Related work

Project planning under uncertainty at the tactical and strategic level has been studied by Adler et al. (1995, 1996). Their research focuses on controlling the number of ongoing projects in an organization and allows estimating the average time spent on a single project. Kavadias and Loch (2004) study project management under uncertainty with respect to project selection and prioritization. The models of Adler et al. (1995, 1996) and Kavadias and Loch (2004), however, do not produce operational scheduling decisions, such as when to initiate or terminate each project phase or individual project activities, which is the goal of this paper.

The literature on *deterministic* operational project scheduling is vast, and contains numerous methods and algorithms to derive project schedules that minimize the project's duration, maximize the project's net present value, minimize the cost of resources, or deal with a host of other objective functions. These algorithms are designed to cope with technological dependencies between the project's activities, modeled in the form of precedence constraints, and other types of constraints, including generalized precedence relations, resource constraints, time/cost and time/resource trade-offs, multiple activity execution modes, and many other characteristics. For recent comprehensive overviews of the literature, we refer to Demeulemeester and Herroelen (2002) and Neumann et al. (2003). The incorporation of *uncertainty* in project planning and scheduling has also resulted in numerous research efforts, particularly focusing on uncertainty in the activities' duration or their cost. The PERT problem has been studied for multiple decades; for a survey see Elmaghraby (1977) and Adlakha and Kulkarni (1987). The consideration of resource constraints has only recently been tackled by, among others, Jørgenson (1999), Stork (2001) and Leus (2003). None of these models, however, incorporate technological uncertainty in the form of stochastic-success activities.

The issue of parallel versus sequential scheduling of project activities has been addressed, among others, by Krishnan et al. (1997) and Eppinger et al. (1994), and is closely related to the topic of concurrent engineering, a systematic approach to the integrated, concurrent design of products and their related processes, including manufacturing and support (Hill, 2002). Hoedemaker et al. (1999) provide some theoretical evidence as to why there are limits to the benefits of parallelization. Parallel (redundant) development of alternative technologies is studied in Abernathy and Rosenbloom (1969), Bard (1985) and Krishnan and Bhattacharya (2002), and a generic representation of multi-stage R&D problems is provided in Lockett and Gear (1973). Zemel et al. (2001) focus on the optimal timing of support activities for R&D tasks of variable length. In Luh et al. (1999), project failure of time-critical projects is equated with not meeting the established target dates, and the scheduling of a set of projects with a number of tasks with uncertain durations, due to design iterations, is undertaken in order to optimize the weighted sum of project earliness and tardiness and the opportunity costs of

failed projects. In our model, we lift the limiting assumption that R&D projects are limited to a single uncertain activity or sequential R&D stages only. Although this is the model commonly found in the literature, it is a major shortcoming for modeling real-life R&D projects because these typically exhibit a network structure, allowing several research tasks to be conducted in parallel. Additionally, our formulation allows modeling R&D projects with inclusion of their ancillary activities such as production and marketing. Pharmaceutical companies, for instance, systematically incur marketing and plant-equipment expenses before the research is completed and success is guaranteed; our model can provide a better understanding of why and when to do so.

The remainder of this paper is organized as follows. Section 3 presents an introductory problem description by means of a real-life example from the pharmaceutical sector. A detailed problem formulation of the *R&D Project Scheduling Problem* (*RDPSP*) and an evaluation of its properties are given in Section 4. In Section 5, we provide a general overview of a branch-and-bound algorithm that is capable of solving the *RDPSP* to optimality, and we present several upper bounds in Section 6. Branching and fathoming details are discussed in Section 7, and Section 8 discusses how the risk preferences of the decision maker can be incorporated. In Section 9, we present computational tests demonstrating the capabilities of our procedure. Finally, a discussion of the results and managerial insights is given in Section 10, with a summary and an outlook on further research in Section 11.

3. An Example

We present an example of a pharmaceutical project, initiated by a biotech company based in Cambridge, England. The pharmaceutical drug development process is heavily regulated, and is monitored in the US by the Food and Drug Administration (FDA). The drug development and review process typically follows four main stages: basic research, pre-clinical, clinical and FDA review, with the clinical stage subdivided in Phase I, II and III. Each clinical substage contains a number of tasks that are repeated several times, each time increasing in duration. The production facilities are developed concurrently with the clinical studies. The entire process takes 12 years on average with total development costs on average of $900 million (DiMasi et al., 2003).

The project was started in 2001 with an expected market launch in late 2007, assuming that the product makes it successfully through all the stages. At the time of this writing, all activities prior to the clinical stage have been successfully performed, and the company is developing a project plan for the clinical development and launch of the product. The total remaining duration of the project is approximately five years, for a total cost of approximately £15 million (all data are disguised). For the purpose of this paper, we have simplified the project plan, which contains more than 300 activities, by identifying natural task groupings, yielding the following aggregate project network

structure. More details on the project can be found in Crama et al. (2004). Phase III in this project is subdivided in three runs of toxicological studies on animals, referred to as "Tox" in Figure 1, and medical studies on humans, referred to as "Med". The remaining activities in Phase III have been grouped in two tasks named "Other", which include manufacturing of the product, chemical product analysis and pharmacological studies. The project also includes the ancillary agronomical task ("Agro"). Each medical study has to be preceded by its corresponding toxicology study. The toxicology studies, however, do not require the results from the previous medical study. Some toxicology and medical studies are dependent on the "Other" activities in the network. The agronomical activity can be scheduled freely.

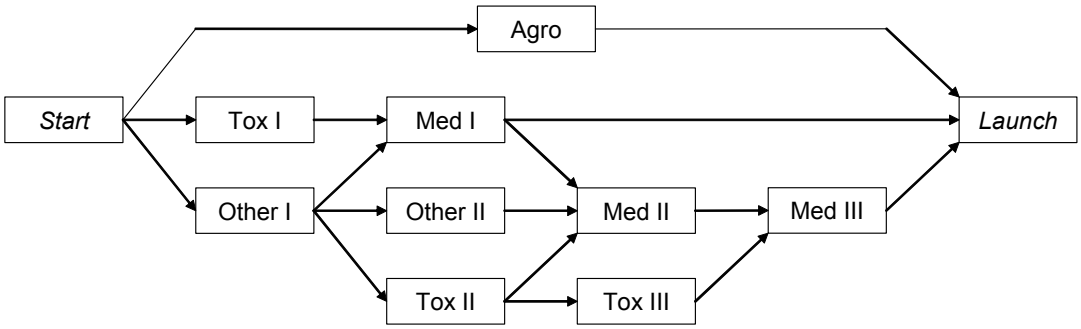


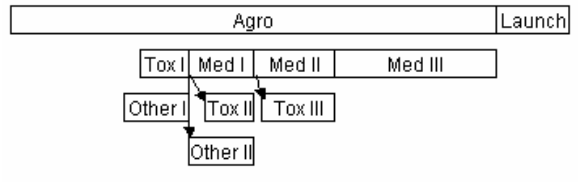
Figure 1. Precedence network for the example project

Table 1 gives, for each activity group, the total development cost, the duration and the probability of technical success (PTS). Cash flows are computed as the sum of the development cost of the subtasks in each group, and durations as the duration of the subproject comprised of the subtasks. The project has an estimated overall PTS of 16.2%. If successful, the net present value of net sales equals £300 million. For this example, we use a discount rate of 1% per month.

Table 1. Project data (disguised)

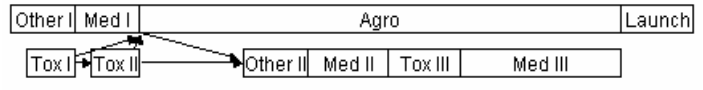
Task	cash flow	duration (months)	probability of technical success
Agro	–£12,000,000	60	100%
Tox I	–£300,000	6	75%
Other I	–£1,000,000	8	100%
Med I	–£200,000	8	80%
Other II	–£300,000	8	100%
Tox II	–£100,000	6	75%
Med II	–£200,000	10	80%
Tox III	–£700,000	9	75%
Med III	–£400,000	20	60%
Launch	£300,000,000	-	-

While developing a schedule for this project, several considerations are in order. If all activities are carried out as soon as possible, the revenues of the project, if successful, are received as soon as possible, resulting in a high present value. On the other hand, development costs are also incurred early on. A better option is to execute the project according to the late-start schedule as determined by the Critical Path Method (CPM). This schedule is pictured in the first schedule of Figure 2 and results in an expected net present value or eNPV of approximately £13 million. Alternatively, we can schedule the activities carrying technical risk in series, thereby avoiding unnecessary expenditures when one of the activities fails. One such schedule is depicted in Figure 2, with an eNPV of approximately £10 million; we notice that project duration may become prohibitively large. Finally, a schedule allowing for a partial overlap of R&D activities is also shown in Figure 2, yielding an eNPV of approximately £16 million, which can be shown to be the highest value achievable. Finding such an optimal schedule is the objective of the algorithms that will be presented in this paper.



(a) CPM late-start schedule

(b) Serial schedule



(c) Optimal schedule

Figure 2. Project schedules; the arrows represent information flows that are not implied by the original precedence relations

The probability distributions of the project's NPV for each of the three schedules are depicted in Figure 3. Clearly, the different schedules exhibit very different risk profiles. The series schedule is conservative and minimizes the downside risk, since it makes full use of the embedded abandonment options, but the total project execution time is maximal. On the other extreme, a CPM schedule results in a large downside risk, compensated for by an earlier launch date, yielding a higher upside potential. In between these two extremes, we find the optimal schedule, which strikes a balance between timeliness of project launch and cost minimization.

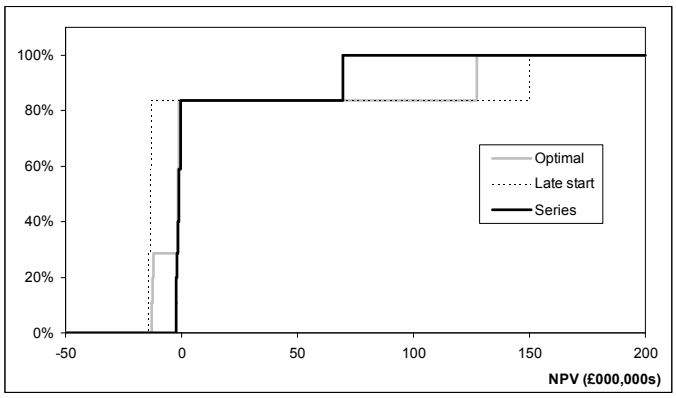


Figure 3. Cumulative distribution function of project NPV.

4. Problem formulation and properties

4.1. Problem formulation and notation

The objective of the R&D Project Scheduling Problem or *RDPSP* is to maximize the expected Net Present Value (eNPV) of the project by constructing a project schedule specifying when to execute each activity. Because of the technical uncertainty inherent to each task, the final project payoff is only achieved when all activities are successful, and an option to abandon the project can be exercised at any time. We focus on the case where all activity cash flows during the development phase are negative, which is typical for R&D projects. The abandonment options imply that each activity will only be started if all the activities scheduled to finish earlier have a positive outcome. Therefore, in the objective function, the activity cash flows are weighted by the probability of joint success of all its scheduled predecessors. We make abstraction of resource constraints and duration uncertainty, and consider the probabilities of technical success of the different tasks as independent.

The following parameters will be used throughout the remainder of this paper:

N	set of project activities $\{0, 1, ..., n\}$
c_i	cash flow of activity $i \in N \setminus \{n\}$, non-positive integer
$c_n(t)$	end-of-project payoff function, non-increasing in time t, integer values
d_i	duration of activity $i \in N$, non-negative integer
p_i	probability of technical success of activity $i \in N$
r	continuous discount rate
A	set of directed activity pairs representing technological precedence relations; A is a (strict) partial order on N, i.e. an irreflexive, antisymmetric and transitive relation
N^*	$\{\{i,j\}: i,j \in N, i \neq j\}$: the set of all unordered activity pairs
$\overline{A}$	$\{\{i,j\} \in N^*: (i,j) \notin A \wedge (j,i) \notin A\}$: the unordered activity pairs that are not comparable according to order relation A; activity pairs in $\overline{A}$ can be executed concurrently

Without loss of generality, we assume activity 0 to be a dummy representing project initiation, with c_0=0, d_0=0 and p_0=1, and $(0,i)\in A$ for all activities $i\in N \setminus \{0\}$ without other predecessors. Activity n represents project completion and is a direct or transitive successor of all other activities. The choice for $c_n()$ to be non-increasing in time is appropriate for most innovative projects: the earlier a new product enters the market, the longer it can benefit from a monopoly position and first-mover advantages, or the longer it can exploit a patent. Activities $N \setminus \{0,n\}$ are referred to as *intermediate* activities. The decision variables are the following:

s_i starting time of activity i; starting-time vector **s** is a schedule

Values for the following variables are associated with a given schedule **s**:

x_{ij} = 1 if $s_i + d_i \le s_j$, = 0 otherwise, for $\{i,j\} \in \overline{A}$. The values x_{ij} are gathered into vector **x**

q_i probability that all activities scheduled to finish no later than s_i will succeed, i.e. probability that activity i will be initiated

The *RDPSP* is now formulated as follows:

$$\text{Max} \qquad g(\mathbf{s}) = \sum_{i=1}^{n-1} q_i c_i e^{-rs_i} + q_n c_n(s_n) e^{-rs_n} \tag{1}$$

subject to

$$s_i + d_i \le s_j \qquad \forall (i,j) \in A \tag{2}$$

$$s_i + d_i \le s_j + T(1 - x_{ij}) \qquad \forall \{i,j\} \in \overline{A} \tag{3}$$

$$q_i = \left(\prod_{(j,i)\in A} p_j \right) \left(\prod_{\{i,j\}\in \overline{A}} (1-(1-p_j)x_{ji}) \right) \qquad \forall i \in N \tag{4}$$

$$x_{ij} \in \{0,1\} \qquad \forall \{i,j\} \in \overline{A} \tag{5}$$

$$s_i \ge 0 \qquad \forall i \in N \tag{6}$$

with $T=\sum_{i\in N} d_i$. The objective function (1) is the eNPV, which is a non-regular measure of performance because starting activities as early as possible is not necessarily optimal. Constraint set (2) imposes the technological precedence constraints. Equation (3) translates starting-time relations between activities that are incomparable with respect to A into values for the 'information flow' variables x_{ij}: the outcome of an uncertain activity delivers useful information for other activities since a failure allows to abandon the project without investing in the remaining tasks. Effectively, equation (3) enforces $s_i + d_i > s_j \Rightarrow x_{ij}$=0. The reverse need not be imposed explicitly since the intermediate project activities have negative cash flows, so that it is always beneficial to the objective function to reduce q_j, i.e. to set x_{ij}=1. A value for probability q_j corresponds with a choice for the information flows x_{ij}, based on equation (4).

4.2. Properties

Instead of using vector **x**, we can equivalently capture information-flow decisions by a relation E on N, defined by $x_{ij}=1 \Leftrightarrow (i,j)\in E$, reflected in the one-to-one function $\chi : \mathbf{x} = \chi(E)$. Each starting-time vector **s** defines a relation $\Sigma(\mathbf{s})$ on N: $(i,j)\in\Sigma(\mathbf{s}) \Leftrightarrow s_i + d_i \le s_j$. A schedule **s** implies the value $\chi(\Sigma(\mathbf{s}))$ for **x**. A relation E corresponds to a feasible schedule **s** (meaning that $\exists \mathbf{s}\in\mathbb{R}^{n+1}$: $\Sigma(\mathbf{s})=E$) if and only if $A \subseteq E$ and E is an order relation; this second condition requires that $G(N,E)$ be acyclic. Once the information-flow decision E is made, determining an optimal-eNPV schedule **s** boils down to scheduling the activities subject to the precedence constraints contained in E. Since we assume that all intermediate cash flows are non-positive, each activity can be scheduled to end at the earliest starting time of its successors in E. Choice $s_0=0$ then corresponds with a unique schedule referred to as $\phi(E)$. In this paper, we focus attention on the search for order relation E with $A \subseteq E$ and which maximizes $g(\phi(E))$, with objective function $g()$ as defined in (1). Note that when the optimal eNPV is negative, the choice $s_0=\infty$, i.e. rejecting the project, is optimal for model (1)-(6); this conclusion can be drawn at the end of the search procedure.

Theorem 1. *If $r=0$ and the project payoff is a constant value ($c_n(t)=c_n$) then an optimal schedule exists for which E is a complete order on N.*

The proofs of the theorems appear in the appendix. The special case without initial precedence constraints ($A=\varnothing$) can be solved in polynomial time:

Theorem 2. *If $r=0$, $A=\varnothing$ and the project payoff is constant ($c_n(t)=c_n$), then each optimal schedule for which E is a complete order, sequences the activities in non-increasing order of $c_i / (1-p_i)$, and each schedule that sequences the activities in non-increasing order of $c_i / (1-p_i)$ is optimal.*

We refer to quantity $c_i / (1-p_i)$ as the *cost-information ratio* of activity i. The result in Theorem 2 is similar to the optimality of the *weighted shortest processing time* rule for one-machine scheduling with total weighted completion time, denoted as $1||\Sigma_j w_j C_j$ (Pinedo, 2002). An efficient polynomial-time algorithm also exists when $G(N,A)$ consists of a number of *parallel chains*. In this case, $N \setminus \{0, n\}$ can be partitioned into K disjoint subsets N_1, N_2, ..., N_K with associated complete order relations A_1, A_2, ..., A_K. We define $[k,l]$ to be the l-th job in chain k, $P(k,l)=\prod_{j=1}^{l} p_{[k,j]}$ and $C(k,l)=c_{[k,1]}+\sum_{i=2}^{l} c_{[k,i]} P(k,i-1)$. Define ρ_k as $\arg\max_{l=1,\ldots,n_k} C(k,l)/(1-P(k,l))$, where $C(k,l)/(1-P(k,l))$ is the cost-information ratio of the chain consisting of the first l jobs of chain k, and $n_k=|N_k|$. We are now ready to propose the following algorithm, which is a generalization of the one appearing in Theorem 2.

Algorithm *PC*. *Construct a complete order on N as follows. Select among all chains k one with highest* $C(k, \rho_k)/(1-P(k, \rho_k))$ *and fill the first available* ρ_k *positions in the complete order with the first* ρ_k *jobs from this chain. If* $\rho_k<n_k$*, consider the remaining part of the selected chain as a separate unselected chain. Iterate this process until all activities of* $N \setminus \{0, n\}$ *have been selected.*

Theorem 3. *If* $r=0$ *and the project payoff is constant* ($c_n(t)=c_n$)*, then Algorithm PC produces an optimal solution to RDPSP with chains in* $O(n^2)$ *time.*

However, the incorporation of precedence constraints imposed by an arbitrary acyclic digraph $G(N,A)$ results in an *NP*-hard problem.

Theorem 4. *RDPSP is NP-hard, even if* $r=0$ *and* $c_n(t)=0$.

5. A branch-and-bound algorithm

In light of the *NP*-hardness of the *RDPSP*, an exact algorithm with better than exponential time complexity is unlikely to exist, and we will devise a branch-and-bound (B&B) algorithm to implicitly enumerate the solution space. Section 5.1 describes the concept of a distance matrix, which is used in Section 5.2 to present a general outline of the B&B algorithm. Throughout the remainder of the text, we assume that $c_n(t)=c_n$, although the algorithms can be adapted to the more general case of time-dependent pay-offs.

5.1. Preliminary concepts

We use the concept of a "distance matrix" as described by Bartusch et al. (1988), collecting information about minimal differences between the starting times of all pairs of activities. The $(n+1)\times(n+1)$-matrix D imposes the following constraints:

$$s_i + D_{ij} \leq s_j \qquad \forall i,j \in N$$

We let the entries in the distance matrix correspond with *all-pairs longest path* distances, in the sense that entry (i,j) represents the length of a longest path from i to j in the complete graph with node set N and distances provided by the matrix. We refer to this property as the *transitive-closure property*. The property can be achieved in $O(n^3)$ time, for instance by means of the Floyd-Warshall algorithm (Lawler, 1976). It can be shown (Bartusch et al., 1988) that a feasible schedule exists iff all $D_{ii}=0$. If $D_{ii}>0$ for some i, the corresponding graph contains a directed cycle with positive length. If $s_0=0$, the earliest possible starting time of activity i is D_{0i}.

We observe that, when $D_{ij} \geq d_i$ for an arbitrary activity pair (i,j), then activity j will always start after activity i has finished, and so $D_{ij} \geq d_i$ implies the possibility of information flow from i to j, denoted as "$i \rightarrow j$". We can also see that the conditions $s_i + d_i > s_j$ and $s_j + d_j > s_i$ jointly imply that i will be executed in parallel with j ("$i \| j$"). Since we work with discrete durations, these conditions are imposed as $D_{ij} \geq -d_j+1$ and $D_{ji} \geq -d_i+1$.

5.2. Outline of the branch-and-bound algorithm

At the root node of the search tree (indexed 0), the distance matrix is initialized as

$$D_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ d_i & \text{if } (i,j) \in A \\ -\infty & \text{otherwise} \end{cases}$$

and the transitive-closure property is enforced. In any node h of the search tree, we partition the set N^* into three disjoint subsets: $N^* = \pi_h \cup \sigma_h \cup \nu_h$, with π_h the set of activity pairs forced to be in parallel according to the distance matrix, σ_h the set of pairs in series, and ν_h the set of pairs that can still be executed both in parallel and in series. Branching continues as long as $\nu_h \neq \emptyset$; a branching decision entails the selection of a set $\{i,j\} \in \nu_h$ and consists of three different branches: (1) $i \rightarrow j$; (2) $j \rightarrow i$; and (3) $i \| j$. These branching options are mutually exclusive and jointly exhaustive. Pursuing a branch means that we update the distance matrix to incorporate the additional constraint(s). Notice that multiple different schedules can correspond with one distance matrix, and also that distance matrix entries are only increased, never decreased, when going from lower- to higher-indexed levels in the search tree (in this way, each search node is more restrictive than its parent node). In any node h of the search tree, the information flows $i \rightarrow j$ are gathered in the order relation E_h, such that $(i,j) \in E_h \Rightarrow \{i,j\} \in \sigma_h$. At initialization, $E_0 = A$.

For each branching operation, updating the distance matrix can be performed in $O(n^2)$ time (Bartusch et al., 1988); the updated matrix in node h is denoted by $D^{(h)}$. The recognition of additional valid parallel relations takes $O(n^2)$ time, recognizing serial relations is embedded in the distance updates and does not add to the $O(n^2)$ time complexity of these updates. Nodes h in the search tree that do not allow a feasible solution are immediately recognized when the distance updates require a change in $D_{ii}^{(h)}$ for some $i \in N$ (see Section 5.1).

Lemma 1. *Any feasible solution in a leaf node h of the search tree ($\nu_h = \emptyset$) for which for at least one activity i, it holds that $\forall (i,k) \in E_h$: $D_{ik}^{(h)} > d_i$, can be discarded without loss of all optimal solutions.*

The proof of the lemma can be found in the appendix. The basic idea of Lemma 1 is that if parallelism constraints are binding for a feasible solution, in the sense that activity starting times would differ if the constraint were removed, then the solution is dominated, because then at least one

activity can be shifted later in time without the artificial constraint. This lemma is the basis for a dominance rule described in Section 7. We underline that parallelism constraints do remain useful for partitioning the search space.

6. Upper bounds

In order to produce an upper bound on the objective-function value in each node of the branch-and-bound tree, we compute bounds on the starting time of each activity. In Section 6.1, we focus on bounding s_n, the project completion time. These bounds are then used to provide an upper bound on the objective function in Section 6.2.

6.1. Bounds on s_n

Since activity n cannot start earlier than $D_{0n}^{(h)}$, this value constitutes a *lower* bound on s_n. A trivial *upper* bound is $\sum_{i\in N\setminus\{n\}} d_i$. A tighter upper bound can be obtained based on the knowledge that in every solution that is not dominated according to Lemma 1, each activity ends exactly when its earliest successor starts, so that the makespan of each non-dominated solution is equal to the sum of the durations of an uninterrupted chain of activities. The length of the longest of such chains can be computed as the length of the longest simple directed path of activity durations from 0 to n in a directed graph G_h constructed as follows. The nodes of G_h are the elements of N, each element $(i,j)\in E_h$ gives rise to arc (i,j), i.e. i must precede j in any chain of activities, and elements $\{i,j\}\in \nu_h$ induce both arcs (i,j) and (j,i). The longest path in G_h needs to fulfill the additional constraint that all precedence constraints in E_h be respected (which is not automatically so), and no activity set in π_h can have both activities in the chain. The resulting path length is referred to as $\mu_0^{(h)}$.

Lemma 2. *The computation of* $\mu_0^{(h)}$ *is an NP-hard problem.*

In light of this complexity status, we will restrict ourselves to upper-bounding $\mu_0^{(h)}$, to which aim we first define the following problem:

MAXIMUM WEIGHT INDEPENDENT SET (MWIS)
Instance: Undirected graph $G(V,C)$, weights w_j for $j\in V$.
Goal: Construct a subset V° of V such that $\sum_{i\in V^\circ} w_i$ is maximized and no two vertices in V° are joined by an edge in C.

As a generalization of MAXIMUM CARDINALITY INDEPENDENT SET (problem GT20, Garey and Johnson, 1979), MWIS is strongly *NP*-hard. For any search node h, the optimal objective-function value of MWIS with $V=N$, $w_i=d_i$ $(i\in V)$ and $C=\pi_h$ equals $\mu_0^{(h)}$, because $G(N,E_h)$ is acyclic and

therefore the graph induced by any subset $S \in N$ is acyclic. Hence, for each independent set S, a linear extension of the partial order induced by S in $G(N,E_h)$ exists. MWIS is an intractable problem but we can obtain an efficiently computable upper bound on its optimum, starting from the following optimization problem and based on Lemma 3.

MINIMUM WEIGHT CLIQUE PARTITION (MWCP)

Instance: Undirected graph $G(V,C)$, weights w_j for $j \in V$.

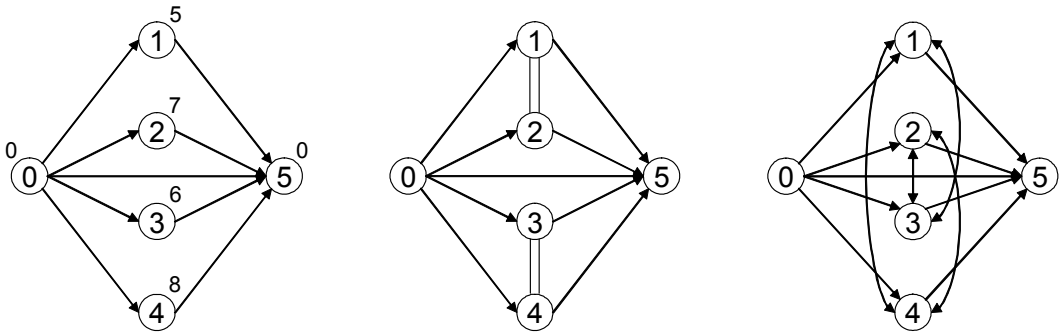
Goal: Partition the vertices of G into disjoint sets V_i such that the subgraph induced by each of these subsets is a complete graph, and such that $\sum_{i=1,\ldots,t} \max_{j \in V_i} \{w_j\}$ is minimized, with t the number of sets in the partition.

Lemma 3. *For undirected graph $G(V,C)$ with weights w_j for $j \in V$ and an arbitrary clique partition into t sets V_i, it holds that $\sum_{i=1,\ldots,t} \max_{j \in V_i} \{w_j\} \geq \sum_{j \in V^\circ} w_j$ for any independent set V° of G.*

MWCP is *NP*-hard since it is a generalization of PARTITION INTO CLIQUES (problem GT15, Garey and Johnson, 1979). From Lemma 3, it follows that any *feasible* solution to MWCP produces an upper bound on the optimal objective-function value of MWIS with the same input parameters. Nevertheless, even the optimal objective value of MWCP may be strictly larger than that of MWIP, in the same way as the *independence number* of a graph is upper-bounded by its *clique partition number* (Busygin and Pasechnik, 2003). The simplest clique partition has $|N|$ cliques (one activity per clique) and results in the trivial bound $\mu_1 = \sum_{i \in N \setminus \{n\}} d_i$ mentioned higher. Its computation takes $O(1)$ time during the course of the algorithm since the summation need only be performed once during the initialization phase. Clique recognition itself being an *NP*-hard problem (problem GT19, Garey and Johnson, 1979), this upper-bound computation seems only little amenable to efficiency enhancements. Our implementation greedily selects cliques of size two and retains the size of the larger of the two durations, according to the following pseudo-code:

```
μ_2^(h) = 0
construct set S=N
for j in S do
    for k in S do
        if {j,k} ∈ π_h then
            add max{d_j,d_k} to μ_2^(h)
            remove j and k from S
            goto next
        endif
    endfor
    add d_j to μ_2^(h)
    remove j from S
next:
endfor
```

The resulting upper bound on s_n is called $\mu_2^{(h)}$, whose computation takes $O(n^2)$ time. The greedy bound $\mu_3^{(h)}$ is constructed similarly, selecting cliques of size (maximum) three, requiring $O(n^3)$ time. We perform multiple runs of the pseudo-code above, each with a different order of examination of the elements of S (in the two "for"-loops); these orders are randomly determined at the initialization phase. A set of experiments revealed that three runs constitute a favorable trade-off between computational effort and the strength of the bound.



(a) (b) (c)

Figure 4. Project networks for the example

We illustrate the different approaches for computing an upper bound on s_n by means of an example. Let n=5 and consider the precedence network $G(N,A)$ of Figure 4(a), with the durations of the activities next to each node. We examine the bounding computations in search node h in which $E_h=A$ (which need not always be the case) and $\pi_h=\{\{1,2\},\{3,4\}\}$, as indicated in Figure 4(b) by the arrows and double lines, respectively. μ_1 is equal to $d_1+d_2+d_3+d_4=26$. $\mu_0^{(h)}$ is the length of the longest simple path from 0 to 5 in graph G_h, which is depicted in Figure 4(c), with an arc with two arrow heads representing one arc in each direction. The path may not contain both activities 1 and 2, and both 3 and 4. One such optimal path is 0-2-4-5 with length 15, which is indeed the tightest bound possible since it is the makespan of e.g. solution $E_h*=A\cup\{(1,3),(1,4),(2,3),(2,4)\}$. An approximation to this quantity is $\mu_2^{(h)}$, which also finds 15 independent of the activity order adopted by the greedy algorithm; the same holds for $\mu_3^{(h)}$.

6.2. Upper bounds on the objective function

Define $g^{(h)}$ as the optimal objective value of model (1)-(6) when we add the constraints that were imposed to reach node h of the search tree. In other words,

$$g^{(h)} = \max_{(\mathbf{q},\mathbf{s}) \text{ respect (2)-(6) and } D^{(h)}} \{ \; q_n c_n e^{-rs_n} + \sum_{i=1}^{n-1} q_i c_i e^{-rs_i} \; \}$$

Since the minimal distance between the starting times of activities i and n is $D_{in}^{(h)}$ in node h, an upper bound on the starting time s_i of intermediate activity i is $s_n - D_{in}^{(h)}$, for a given value of s_n. From Section 6.1, we know that s_n itself takes on a (discrete) value in $[D_{0n}^{(h)} ; \mu]$, with μ any of the upper bounds on s_n developed in that section. Therefore, in the knowledge that $c_i \leq 0$ for $i \in N \setminus \{0,n\}$,

$$g^{(h)} \leq \max_{\substack{t \in [D_{0n}^{(h)};\mu]; \\ \exists \mathbf{s}:(\mathbf{q},\mathbf{s}) \text{ respect} \\ \text{(2)-(6) and } D^{(h)} \text{ and } s_n \leq t}} \{ q_n c_n e^{-rt} + \sum_{i=1}^{n-1} q_i c_i e^{-r(t-D_{in}^{(h)})} \},$$

with the right hand side equal to

$$\max_{t \in [D_{0n}^{(h)};\mu]} e^{-rt} \{ q_n c_n + \max_{\substack{\exists \mathbf{s}:(\mathbf{q},\mathbf{s}) \text{ respect} \\ \text{(2)-(6) and } D^{(h)} \text{ and } s_n \leq t}} \{ \sum_{i=1}^{n-1} q_i c_i e^{rD_{in}^{(h)}} \} \} \tag{7}$$

since q_n is a constant. This leaves us with the evaluation of the quantity

$$\max_{\substack{\exists \mathbf{s}:(\mathbf{q},\mathbf{s}) \text{ respect} \\ \text{(2)-(6) and } D^{(h)} \text{ and } s_n \leq t}} \{ \sum_{i=1}^{n-1} q_i c_i e^{rD_{in}^{(h)}} \}. \tag{8}$$

A lower bound on q_i, the execution probability of activity i, is $\theta_i^{(h)} = \prod_{j \in N: D_{ij}^{(h)} \leq -d_j} p_j$, which leads to an upper bound on (8) that does not depend on t. The resulting bound $g^{(h)}$ is referred to as UB_1, for which the maximum in t is found for the left or right boundary of the allowable interval, depending on the sign of the expression following e^{-rt} in (7). It is interesting to see that in non-dominated leaf nodes, UB_1 equals the actual objective-function value when it is non-negative, and also when it is negative and $\mu = \mu_0$. In addition, we remark that computational effort for determining μ is only required when (8) is larger than $q_n c_n$ in absolute value, so when the incumbent has an objective function below zero. Only in those cases will 'sophisticated' bounds such as μ_2 or μ_3 be useful.

Alternatively, if we relax the constraints with respect to $D^{(h)}$ and t, (8) is the optimal objective value of an auxiliary *RDPSP*-instance for the same set of activities N with precedence graph $G(N,A)$ but with r=0, c_n=0 and intermediate cash flows $c_i' = c_i e^{rD_{in}^{(h)}}$. Solving this instance is still difficult, because *RDPSP* with general precedence constraints, r=0 and c_n=0 is *NP*-hard (Theorem 4). The objective function when the precedence constraints are relaxed (A=∅), however, produces an upper bound to the precedence-constrained case, and this problem can be solved by sequencing the activities in non-increasing order of $c_i'/(1-p_i)$ (Theorem 2). We thus obtain UB_2.

An improvement on UB_2 can be obtained in the knowledge that *RDPSP* with r=0 and precedence constraints in the form of chains can be easily solved (Theorem 3). In a first pass, we greedily (based on the order of addition to E_h) construct a set of chains from E_h and impose only those constraints on the auxiliary problem (i.e. we relax some of the task dependencies so that only chains result). This bound is further improved by noting that for two activities $\{i,j\} \in \pi_h$, the addition of either (i,j) or (j,i) to the possible precedences composing the chains, still results in a relaxation of the original problem: only one of the two distance increases for implementing $i||j$ is then relaxed, instead of both. In our implementation, we add (i,j) if $c_i'/(1-p_i) \geq c_j'/(1-p_j)$, otherwise (j,i). We call the resulting bound UB_3.

7. Algorithmic structure and details

Overall structure of the algorithm. A general overview of the B&B algorithm is given in Figure 5.

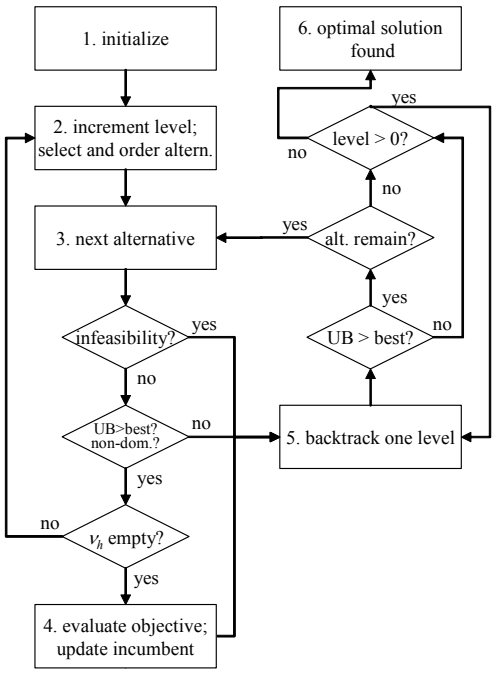


Figure 5. Flow chart of the algorithm

Branching choice. We explore different rules for the selection of an activity pair $\{i,j\} \in \nu_h$ to branch on. As a first possibility, rule 1 selects the first encountered activity pair $\{i,j\}$ in ν_h based on lexicographic ordering of the alternatives. From our experiments we have observed that the 'low-impact' choices in the optimization concern activities with a lot of slack in their starting times. Therefore, we have implemented rule 1 with activity ordering based on (1) the activity index and (2) float values (*increasing* CPM-based total float in $G(N,A)$). The goal of this second option is to make decisions that strongly affect the bounds on lower-indexed levels in the search tree.

Alternatively, we order the candidate activity pairs in decreasing order of a 'pseudo-cost' of insertion, which is an estimate of their true impact. The role of this pseudo-cost is in guiding heuristic decisions in the algorithm, not in generating incumbent solutions or in proving fathomability (Parker and Rardin, 1988). Rule 2 selects $\{i,j\} \in \nu_h$ with highest ratio $c_i/p_j+c_j/p_i$, in an attempt to make the most important decisions first. Rule 3 also tries to select the most influential activity pair $\{i,j\}$ first, by maximizing the difference between the latest ending time of the earliest starting activity (latest start times are given by $D_{0n}^{(h)} - D_{in}^{(h)}$) and the latest start of the other activity. Finally, rule 4 is a criterion that (approximately) minimizes the number of nodes in the search tree: we choose the activity pair that allows removing the most elements from ν_h, summed over its three emanating branches. An estimate of the number of elements removed by alternative $i \rightarrow j$ is $\#\{k \in N: ((j,k) \in E_h \wedge$

$(i,k)\notin E_h$) $\vee$ ($(k,i)\in E_h \wedge (k,j)\notin E_h$)}; an estimate of the effect of $i\|j$ is $\#\{k\in N$: ($\{j,k\}\in\pi_h \wedge \{i,k\}\notin\pi_h$) $\vee$ ($\{k,i\}\in\pi_h \wedge \{k,j\}\notin\pi_h$)}. Ties are resolved either lexicographically or based on the activity float values as discussed earlier.

Branching order. We examine two different approaches with respect to the branching order, i.e. the order in which the three branches $i\rightarrow j$, $j\rightarrow i$ and $i\|j$ are explored once a branching choice $\{i,j\}$ has been made. One possibility is to adhere to a *fixed* branching order; the actual order in this case turns out not to be decisive for algorithmic performance, we implement (1) $i\rightarrow j$, (2) $j\rightarrow i$ and (3) $i\|j$. The second option is to use a *variable* order, in which we first select the branch that is compatible with the current best known solution: if $s_i + d_i \le s_j$ in this schedule, we first explore $i\rightarrow j$, then $i\|j$ and finally $j\rightarrow i$. If i and j overlap in the incumbent, we first explore $i\|j$; the second alternative is $i\rightarrow j$ if $s_i \le s_j$.

Dominance rule. Based on Lemma 1, we implement a *dominance rule* as follows: if an activity $i\neq 0,n$ exists for which $\{k\in N:\{i,k\}\in\nu_h\}=\varnothing$ and $\forall k\in N|(i,k)\in E_h$: $D_{ik}^{(h)}>d_i$, then the current search node can be fathomed. The rule is valid because distance-matrix entries can only increase, never decrease, when descending the search tree. The test itself can be run in $O(n)$ time, since we dynamically maintain count of the cardinality of sets $\{k\in N:\{i,k\}\in\nu_h\}$, $\{k\in N:(i,k)\in E_h\}$ and $\{k\in N: D_{ik}^{(h)}>d_i\}$.

A heuristic stand-alone procedure. We propose a heuristic that examines a set of solutions starting from $\phi(A)$, the latest-start schedule based only on the technological precedence constraints, and then gradually appending precedence constraints until a full order is obtained; each intermediate solution is evaluated and the best one retained. The procedure can be described in pseudo-code as follows. The output of this heuristic is used at the initialization phase of the B&B algorithm to produce a good lower bound.

```
s_best := φ(A); E := A
construct full order R extending A, sequencing incomparable activities in
        non-increasing order of c_i/(1−p_i)
for d = n−2 downto 1 do
        S is the set of ordered activity pairs (i,j) for which the difference between the
                rank order of i and j in R equals d and j comes after i in R
        order the elements (i,j)∈S in decreasing −c_j/p_i
        for ordered (i,j)∈S do
                E := E∪{(i,j)}
                s := φ(E)
                if g(s) > g(s_best) then
                        s_best := s
                endif
        endfor
endfor
```

8. Considering risk preferences

Maximizing the expected NPV does not preclude actual project realizations from resulting in higher or lower NPV values. In order to evaluate the risk profile associated with a project schedule, a representation of all possible NPV realizations, together with the probability mass function (pmf) value of each realization, would be desirable. In the literature on project networks with stochastic activity durations, it is shown (Hagstrom, 1988; Möhring, 2001) that even with independent processing-time distributions, the determination of a single point of the cumulative distribution function (cdf) of the project completion time is #*P*-complete, and thus in particular *NP*-hard. As noted by Adlakha and Kulkarni (1989), the difficulty arises from two sources: (1) the number of paths grows exponentially in the number of activities, and (2) even when the activity durations are independent, the path lengths are generally dependent, as there are activities common to more than one path.

Fortunately, our setting of stochastic-success activities does not result in the same difficulties. In spite of the fact that $O(2^n)$ different realizations are possible of success or failure of the individual activities, the knowledge that activity failure leads to immediate project termination admits efficient determination of the pmf of the NPV for an arbitrary schedule. With each schedule **s**, we associate a set $\delta(\mathbf{s})$ of decision points corresponding with the (intermediate) activity start and finish times: $t \in \delta(\mathbf{s}) \Leftrightarrow \exists i \in N \setminus \{0,n\}: t=s_i \vee t=s_i+d_i$. The following procedure correctly determines the NPV-pmf of **s**, denoted $f_{\mathbf{s}}(\cdot)$, and its expected NPV $g(\mathbf{s})$; it can be implemented in $O(n\log n)$ time. For easy access, $\delta(\mathbf{s})$ is conceived and ordered as a multi-set (which is *not* explicitly taken into account in the code description below). A bifurcation of probability mass occurs at each decision point at which fallible activities ($p_i<1$) end.

cumprob=1; *cumcost*=0
$f_{\mathbf{s}}(\cdot)=0$; $g(\mathbf{s})=0$
for increasing t in $\delta(\mathbf{s})$ **do**
  if $\exists i \in N: t=s_i+d_i$ **then**
    $tprob=\prod_{i \in N \mid t=s_i+d_i} p_i$
    if ($tprob < 1$) **then**
      $f_{\mathbf{s}}(cumcost) := f_{\mathbf{s}}(cumcost) + cumprob*(1-tprob)$
      $g(\mathbf{s}) := g(\mathbf{s}) + cumcost*cumprob*(1-tprob)$
      $cumprob := cumprob*tprob$
    endif
  endif
  for all $i \in N \mid s_i=t$ **do**
    $cumcost := cumcost + c_i . e^{-rs_i}$
  endfor
endfor
$cumcost := cumcost + c_n . e^{-rs_n}$
$f_{\mathbf{s}}(cumcost) := f_{\mathbf{s}}(cumcost) + cumprob$
$g(\mathbf{s}) := g(\mathbf{s}) + cumcost*cumprob$

The NPV-pmf can be used by the decision maker to evaluate the downside risk, e.g. the probability that the NPV is lower than or equal to a threshold value, or the upside potential, e.g. the probability that NPV is larger than or equal to a threshold. This gives the decision maker a number of additional options, (a) it allows for the specification of a constraint on downside risk and/or upside potential, which can imposed during the search for schedules with maximum NPV, and (b) the approach permits to generate the efficient frontier showing the trade-off between return and risk.

9. Computational experiments

We have performed a series of computational experiments using randomly generated test problems in order to examine the behavior of the B&B algorithm.

9.1. Experimental setup

Random test sets have been generated for various values of n using the random network generator *RanGen* (Demeulemeester et al., 2003). Each dataset contains 20 instances for each of the values 0.25, 0.50 and 0.75 of the network-shape parameter *order strength* (*OS*; Mastor, 1970), resulting in 60 instances per set. *OS* is the number of comparable intermediate activity pairs divided by the maximum number $(n-1)(n-2)/2$ of such pairs, and is a measure for the closeness to a linear order of the technological precedence constraints in A. For all experiments, we set $r = 0.05$. Cash flows for each activity in $N \setminus \{0,n\}$ are generated as independent realizations of a discrete uniform random variable on [−50 ; 0]; durations for these activities are discrete values in [1 ; 15] and success probabilities are randomly chosen between 80% and 100%. Dummy start activity 0 has $d_0=c_0=0$ and $p_0=1$. The end-of-project payoff value c_n is a discrete value randomly selected from the interval $[-0.5a\ ;\ -2a]$ with

$$a= (1/q_n^{(0)})\left(\sum_{i=1}^{n-1} c_i q_i^{(0)} \exp(0.05 D_{in}^{(0)})\right),$$

with distance matrix $D^{(0)}$ based on the initial order relation A, and probabilities $q_i^{(0)}$ based on starting times $(D_{0n}^{(0)} - D_{in}^{(0)})$. Note that when $c_n \geq -a$ the optimal project's NPV is non-negative.

The B&B algorithm was coded in C using Microsoft Visual C++ 6.0, and the experiments were run on a Dell OptiPlex GX240 PC with an Intel 1,500 MHz processor and 256 MB RAM, equipped with Windows 2000. A time limit of 150 seconds is imposed on the running time of the algorithms.

9.2. Comparison of solutions

When comparing different algorithms, we need a measure of the quality of the solutions generated. Since the objective-function value can be either positive or negative, comparing the quality of two different schedules $\mathbf{s}_1$ and $\mathbf{s}_2$ is not trivial. We use two different measures:

(1) improvement $I(\mathbf{s}_1,\mathbf{s}_2) = (g(\mathbf{s}_2) - g(\mathbf{s}_1)) / \min\{|g(\mathbf{s}_1)|,|g(\mathbf{s}_2)|\}$

The improvement function measures the extent to which the objective function is improved by moving from solution $\mathbf{s}_1$ to $\mathbf{s}_2$. We report the average improvement across all instances for which $g(\mathbf{s}_1)$ and $g(\mathbf{s}_2)$ have the same sign, and indicate with a separate figure the number of instances with different sign, since in that case $I(\mathbf{s}_1,\mathbf{s}_2)$ is not meaningful.

(2) scaled distance $\Delta(\mathbf{s}_1,\mathbf{s}_2) = (g(\mathbf{s}_2) - g(\mathbf{s}_1)) / |g(\mathbf{s}_2) - g(\mathbf{s}_{LSS})|$

$\mathbf{s}_{LSS} = \phi(A)$, the initial late-start schedule, is used as a benchmark solution. If $g(\mathbf{s}_2) \geq g(\mathbf{s}_1) \geq g(\mathbf{s}_{LSS})$ then $\Delta(\mathbf{s}_1,\mathbf{s}_2)$ maps into [0 ; 1]; when $\mathbf{s}_2=\mathbf{s}_{LSS}$, we define $\Delta(\mathbf{s}_1,\mathbf{s}_2)=0$. $\Delta(\mathbf{s}_1,\mathbf{s}_2)$ close to 0 means that the additional benefit of $\mathbf{s}_2$ over $\mathbf{s}_1$ is rather limited, whereas $\Delta(\mathbf{s}_1,\mathbf{s}_2)$ close to 1 indicates that $\mathbf{s}_2$ is considerably better than $\mathbf{s}_1$.

The improvement function $I()$ is sensitive to whether or not the objectives function values are close to zero, whereas $\Delta()$ yields a more stable measure but is dependent on the quality of heuristic solution $\mathbf{s}_{LSS}$.

9.3. Computational efficiency

For the dataset with n=15, Table 2 shows the successive improvements in the efficiency of the B&B algorithm when components are added. The table shows the average CPU time in seconds, the average number of nodes in the search tree, the number of instances solved to guaranteed optimality (within 150 seconds), and several quality measures. The base case in the table relates to the following parameter settings: lexicographic branching choice (rule 1) using index order, fixed order branching, no dominance rule, no initial solution, and makespan upper bound μ_1. $\mathbf{s}_{(i)}$ is the output of the procedure run in setting (i). $\mathbf{s}_{HEU}$ refers to the schedule produced by the heuristic described in Section 7. For comparison purposes, the values for CPU time and number of nodes are expressed as a fraction of the value for setting (13), which is the most efficient version of the algorithm and which contains the initial lower bound, UB_1, branching rule 1 based on float values, variable branching order, the dominance rule, and μ_2 and μ_3.

Table 2. Successive improvements in the efficacy and efficiency of the B&B procedure for n=15.

	CPU	nodes	opt (/60)	$I(\mathbf{s}_{HEU},\mathbf{s}_{(i)})$	$I(\mathbf{s}_{LSS},\mathbf{s}_{(i)})$	$\Delta(\mathbf{s}_{HEU},\mathbf{s}_{(i)})$	$I(\mathbf{s}_{(i)},\mathbf{s}_{(13)})$
(1) *base* + UB_1	62,807%	93,478%	29	–0.317 (21)	–0.088 (22)	–0.543	0.555 (22)
(2) = *base* + UB_1 + *init. LB*	557%	684%	50	0.203 (0)	0.320 (1)	0.541	0.083 (1)
(3) = *base* + UB_2 + *init. LB*	116,447%	211,824%	21	0.092 (0)	0.196 (1)	0.219	0.197 (1)
(4) = *base* + UB_3 + *init. LB*	79,912%	85,742%	23	0.092 (0)	0.197 (1)	0.247	0.196 (1)
(5) = (2) + rule 2, *index*	4,368%	4,832%	44	0.221 (0)	0.350 (1)	0.533	0.062 (1)
(6) = (2) + rule 3, *index*	8,234%	9,497%	43	0.205 (0)	0.322 (1)	0.548	0.081 (1)
(7) = (2) + rule 4, *index*	625%	706%	50	0.203 (0)	0.320 (1)	0.541	0.083 (1)
(8) = (2) + rule 1, *float*	538%	650%	50	0.204 (0)	0.321 (1)	0.542	0.083 (1)
(9) = (2) + rule 4, *float*	633%	693%	50	0.204 (0)	0.321 (1)	0.542	0.083 (1)
(10) = (8) + *var. br. order*	407%	521%	51	0.266 (1)	0.389 (2)	0.612	0.013 (0)
(11) = (10) + *domin. rule*	191%	228%	51	0.267 (1)	0.391 (2)	0.612	0.012 (0)
(12) = (11) + μ_2	106%	107%	52	0.280 (1)	0.412 (2)	0.614	0.006 (0)
(13) = (12) + μ_3	100%	100%	52	0.290 (1)	0.427 (2)	0.615	0.000 (0)

9.4. Algorithmic performance for various problem sizes

Table 3 presents results for different problem sizes, with **s*** the output schedule of the B&B algorithm. The rightmost column measures the increase in the makespan of the output schedule $s_n(\phi(E))$ relative to the LSS schedule. The results reveal that the quality of the B&B algorithm is considerably higher than that of the heuristic. We also observe that the computational effort is inversely related to *OS*. This is an intuitive result, since the number of 'undecided' activity pairs in ν_0 is higher for lower *OS* values. Finally, since the final makespan is often higher than the critical-path length (especially for *OS*=0.25), we observe that there are limits to the benefits of parallelization.

Table 4 examines the performance of the B&B procedure for various time limits with n=30; $\mathbf{s}_{(i)}$ is the output of the procedure run with time limit i. A time limit of zero means that the actual branching procedure is never entered so that the initial heuristic solution is the output of the algorithm. When a time limit of 1,500 seconds is imposed, four more instances are solved to guaranteed optimality compared with 150 seconds, and the quality of the solutions improves slightly. The percentage difference between the solutions in these two settings is 1.07%, and an improvement in objective-function value is achieved for 26 out of the 60 instances.

Table 3. Performance of the truncated B&B algorithm for various values of n.

OS	n	CPU	opt (/20)	nodes	$I(\mathbf{s}_{\text{HEU}},\mathbf{s}^*)$	$I(\mathbf{s}_{\text{LSS}},\mathbf{s}^*)$	$\Delta(\mathbf{s}_{\text{HEU}},\mathbf{s}^*)$	$s_n(\phi(E))/s_n(\phi(A))$
	9	0.08	20	11,396	0.360 (1)	0.607 (1)	0.47	1.23
	12	15.0	19	1,707,429	0.706 (1)	1.073 (2)	0.45	1.19
	15	38.5	15	3,378,899	0.201 (1)	0.606 (1)	0.59	1.25
0.25	18	49.0	17	3,642,441	0.079 (0)	0.196 (0)	0.63	1.23
	21	147.4	1	8,871,451	0.236 (1)	1.123 (1)	0.58	1.38
	24	150.0	0	7,520,300	0.169 (0)	0.851 (1)	0.36	1.87
	27	150.1	0	6,340,850	0.056 (0)	0.348 (1)	0.49	1.69
	30	150.1	0	5,488,850	0.232 (0)	1.184 (0)	0.55	1.69
	9	0.01	20	414	0.058 (0)	0.130 (0)	0.36	1.08
	12	2.53	20	279,007	0.173 (0)	0.336 (0)	0.48	1.11
	15	24.1	17	2,005,257	0.391 (0)	0.552 (0)	0.71	1.06
0.50	18	40.5	15	2,731,624	0.100 (0)	0.123 (1)	0.73	1.07
	21	64.6	13	3,890,347	0.253 (0)	0.286 (0)	0.76	1.03
	24	146.4	1	7,035,465	0.108 (0)	0.256 (0)	0.61	1.38
	27	140.7	2	5,952,657	0.083 (0)	0.333 (0)	0.66	1.48
	30	150.1	0	5,648,750	0.319 (0)	0.221 (2)	0.65	1.14
	9	0.01	20	46	0.020 (0)	0.036 (0)	0.21	1.02
	12	0.02	20	1,720	0.084 (0)	0.684 (0)	0.34	1.06
	15	0.70	20	53,274	0.274 (0)	0.111 (1)	0.54	1.04
0.75	18	30.1	16	1,867,598	0.127 (1)	0.154 (1)	0.74	1.04
	21	37.8	15	1,920,565	0.046 (0)	0.218 (0)	0.69	1.02
	24	24.6	17	1,069,091	0.024 (0)	0.092 (0)	0.69	1.01
	27	85.2	10	3,636,001	0.047 (0)	0.101 (0)	0.69	1.08
	30	144.0	2	5,188,602	0.061 (0)	0.311 (0)	0.62	1.02

Table 4. Performance of the truncated B&B algorithm for n=30, for varying time limits (in seconds).

time limit	opt (/60)	nodes	$I(\mathbf{s}_{\text{HEU}},\mathbf{s}_{(i)})$	$I(\mathbf{s}_{\text{LSS}},\mathbf{s}_{(i)})$	$\Delta(\mathbf{s}_{\text{HEU}},\mathbf{s}_{(i)})$	$I(\mathbf{s}_{(i)},\mathbf{s}_{(1500)})$
0	0	0	0 (0)	0.384 (2)	0	0.219 (0)
1	0	39,033	0.161 (0)	0.528 (2)	0.53	0.046 (0)
10	0	371,567	0.181 (0)	0.557 (2)	0.57	0.030 (0)
60	0	2,210,450	0.196 (0)	0.575 (2)	0.60	0.018 (0)
150	2	5,442,067	0.204 (0)	0.584 (2)	0.61	0.011 (0)
1500	6	51,355,273	0.219 (0)	0.601 (2)	0.62	0.000 (0)

10. Discussion

In this section, we analyze the properties of the optimal solution to the *RDPSP* by varying the main parameters in the example project presented in Section 3. We study the total project duration and the optimal schedule, and formulate guidelines for scheduling in R&D environments.

10.1. Limits to the benefits of parallelization

We observed that the optimal schedule for the example R&D project is different from the CPM late-start schedule, and that its duration exceeds the CPM duration; a similar conclusion was drawn with respect to the test set instances after examination of Table 3. In line with Hoedemaker et al. (1999), we observe that there are limits to the benefit of parallelization in R&D projects. This contrasts with projects without technical uncertainty, for which a CPM late-start schedule is optimal. Clearly, in the case of project success, the present value of the final project payoff decreases with increasing project duration. However, short project duration requires overlapping activities, which increases the expected expenditures. This fundamental difference is due to the fact that in R&D projects, even though activities may not exhibit a technical dependency, each activity releases information on the final success of the project at completion. If activities are in series, the later activities can benefit from the information released by the already completed activities by exercising an abandonment option in case of failure. Therefore, R&D project scheduling requires balancing early project completion with minimizing expected expenditures.

10.2. The impact of the discount rate on project duration

Intuitively, one would expect that the incentive for parallelization increases with increasing cost associated with project delay. We examine this behavior by means of varying the interest rate, representing the time value of money, for a constant project payoff. With zero discount rate, the value of the project payoff is constant over time and the project schedule can take full advantage of information flows: no activity with PTS < 1 will be in parallel with other activities, which leads to maximum total project duration (this insight leads to Theorem 1). As the interest rate goes up, we observe a reduction in the optimal project duration: some activities are overlapped, forfeiting information flows for the sake of earlier project completion. Interestingly, however, from a certain point onwards a further increase in the interest rate induces an increase in the optimal project schedule length. The reason for this phenomenon is that the magnitude of the present value change due to an incremental delay in a cash flow decreases as the interest rate becomes larger, and this effect is more marked for cash flows that occur later in time. As a consequence, cost savings early on in the project

due to higher project duration may more than offset the associated decrease in the present value of the project payoff.

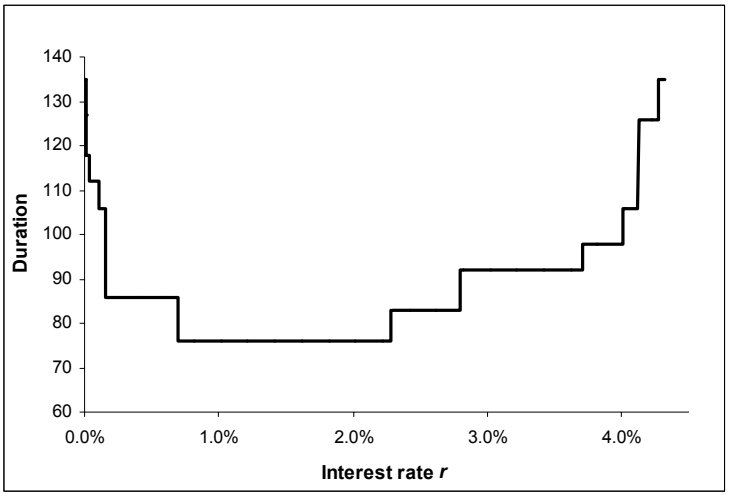


Figure 6. Project duration versus interest rate for the example problem

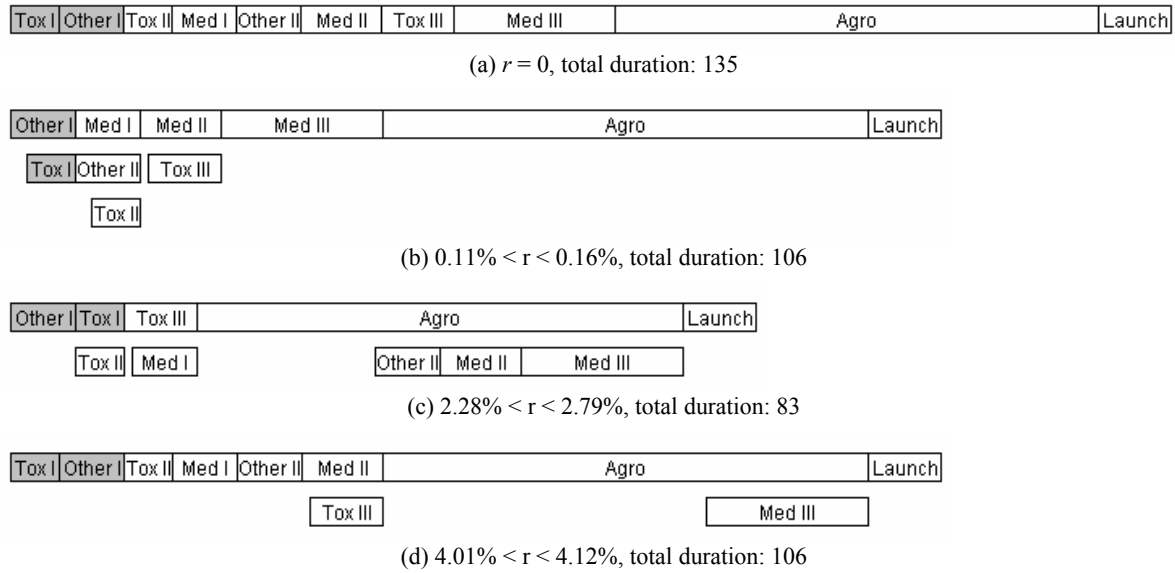
As shown in Figure 6, the optimal project duration is a convex step function of the interest rate for the example problem. The graph also indicates that different scheduling choices are made before and after the zone of minimum duration: the left part of the graph is not a mere mirror image of the right part. In Figure 7, optimal schedules are shown for four different ranges of the discount rate. We observe an initial reduction in the project duration, from case (a) to (b) to (c), followed by an increase, from (c) to (d). The optimal schedule in (d), however, is different from that in (b), although the project duration is the same. We also observe the variable character of the precedence relationship between pairs of activities in the different optima, for instance for the shaded activity pair {Tox I, Other I}. This exemplifies the unpredictable nature of the optimal schedule to the *RDPSP*.

10.3. Guidelines for scheduling R&D projects

The main determinants affecting the optimal timing of an R&D activity, and whether or not it should be scheduled in parallel with other R&D activities, are its PTS, its cost, its duration, and its location in the network.

Based on Theorem 2, we anticipate that activities with low cost and low PTS should generally be scheduled early. This is logical: if activities with relatively higher costs are scheduled later, then the associated cash outflow is discounted, and also reduced in expected value because the project may be terminated sooner. Activities with low PTS values best precede a high number of activities, reducing the expected cash flow of their successors.

In the absence of discounting, the value of the information loss by scheduling two activities i and j in parallel rather sequentially can be quantified as $\max\{(1-p_i)c_j\ ;\ (1-p_j)c_i\}$, which should be more than offset by the gain in project lead time in order for parallelization to make sense. Consequently, prime candidates for parallelization are activities with high PTS *and* low cost. Another determinant for parallelization of an activity is its position in the precedence network: activities are less likely to be performed if they are preceded by many other uncertain activities, reducing the expected value of the information loss. Therefore, if PTS and cost are of the same order of magnitude throughout the network, then parallel execution of activities will preferably occur toward the end of the schedule. This preference is reinforced when the interest rate rises to such an extent that the optimal total project duration becomes a non-decreasing function of the interest rate. At that point, the timing of the cost increase due to parallelization becomes relatively more important than its size, and there is a tendency to keep parallel activities late in the project. This is illustrated in schedules (c) and (d) in Figure 7.



(a) $r = 0$, total duration: 135

(b) 0.11% < r < 0.16%, total duration: 106

(c) 2.28% < r < 2.79%, total duration: 83

(d) 4.01% < r < 4.12%, total duration: 106

Figure 7. Optimal schedules for different interest rates

Based on the foregoing two paragraphs, we conclude that especially activities with high PTS in the later parts of the project should be considered for parallelization. Low-cost activities are generally also good candidates, except when they occur early in the project schedule, which is often the case. We also observe that activities of similar duration, or sequences of activities of similar duration, are more likely to be put together, in order to maximize the time-savings by parallelization for the same information loss.

11. Summary and outlook on further research

We have presented a model and algorithms for scheduling R&D projects to maximize the expected Net Present Value (eNPV) when the project activities have an inherent possibility of failure and when individual activity default causes overall project failure. We have shown that this problem, referred to as the R&D Project Scheduling Problem or *RDPSP*, is *NP*-hard, but presented polynomial-time algorithms for special cases. We have also developed a branch-and-bound algorithm that is able to produce high-quality project schedules. Our results show that, although the present value of the project payoff decreases with increasing project duration, performing certain activities in series rather than in parallel can decrease the development cost of a project, resulting in an overall improvement in the project's eNPV. An optimal project schedule will need to balance information flows between activities against delays in final project payoff: some activities may need to overlap, while others may need to be executed in sequence. In line with the findings of Hoedemaker et al. (1999), we have found that there are limits to the benefits of parallelization.

A number of opportunities for future research exist. The explicit incorporation of resource constraints or duration uncertainty enhances the accuracy with which the R&D planning process is modeled, but introduces an additional level of complexity that may prove to be computationally overwhelming. Another possible extension of our base model is correlated activity success, an inherent characteristic of most R&D projects. Quantifying such correlations may be difficult, however. The model could also be extended by including alternative sets of activities for which success is required for only one set, allowing to model the pursuit of alternative technologies. Finally, the model can be altered by taking into account that some R&D activities can be performed in different ways, e.g. by allocating more or less money, resulting in different success probabilities associated with these multiple activity execution modes.

Appendix

***Proof of Theorem* 1**

When r=0, the objective function equals $\sum_{i=0}^{n-1} q_i c_i + q_n c_n$, with q_n independent of the information-flow decisions. Therefore, each optimal order E minimizes $\sum_{i=0}^{n-1} q_i |c_i|$. Consider any optimal order E_0 that is *not* a complete order. If no such E_0 exists, the theorem holds, otherwise, take an arbitrary activity $k^\circ \in N$ that is incomparable with at least one other element in N according to E_0. The expression to be minimized can now be written as follows, in which $\bar{E}_0$ is the set of unordered incomparable activity pairs according to E_0:

$$\sum_{i=0}^{n-1} q_i |c_i| = q_{k^\circ} |c_{k^\circ}| + \sum_{(i,k^\circ) \in E_0} q_i |c_i| + \sum_{\{i,k^\circ\} \in E_0} q_i |c_i| + \sum_{(k^\circ,i) \in E_0} q_i |c_i|$$

If we extend E_0 to $E_1 = E_0 \cup \{(k^\circ,i) : \{k^\circ,i\} \in \bar{E}_0\}$, the only term changing in the right hand side of the foregoing equation is the third: $\sum_{\{i,k^\circ\} \in E_0} q_i |c_i|$ is multiplied by p_{k°. We conclude that the objective-function value associated with E_1 is at least as good as the value for E_0. Continuing in this way, we can obtain a complete order E^* for which the associated objective-function value is at least as high as for E_0, after at most $(n-1)$ iterations. □

***Proof of Theorem* 2**

Suppose that a complete order E_0 is optimal, by which two activities i and j are scheduled consecutively, $(i,j) \in E_0$, and $c_i / (1-p_i) < c_j / (1-p_j)$. The expected discounted cash flow of activities i and j equals $\left(\prod_{(k,i) \in E_0} p_k\right)(c_i + p_i c_j)$. We now construct a new complete order E_1 as the result of the exchange of positions of i and j in E_0 (a so-called *adjacent pairwise interchange* in scheduling theory). Clearly, the expected discounted cash flow of activities i and j for E_1 equals $\left(\prod_{(k,i) \in E_0} p_k\right)(c_j + p_j c_i)$, while the expected cost of the activities in $N \setminus \{i,j\}$ remains unchanged. Knowing that $(c_i + p_i c_j) < (c_j + p_j c_i)$, we observe that the NPV of E_1 is higher than the NPV of E_0, which contradicts the optimality of E_0. This proves the first part of the theorem.

The second part of Theorem 2 holds because an optimal complete order always exists (Theorem 1), and activities with equal cost-information ratio can interchange positions without effect on the objective function. □

In order to prove Theorem 3, we first derive some preliminary results (Lemmas A1 and A2).

Lemma A1. *If r=0, project payoff is constant, the precedence constraints take the form of chains, and the solution space is restricted to complete order relations in which all activities of each chain are ordered contiguously, then each optimal schedule sequences the chains k in non-increasing order of $C(k,n_k)/(1-P(k,n_k))$, and each schedule with such an order is optimal.*

Proof: A trivial extension of Theorem 2. □

Lemma A2. *Given two chains* (N_1,A_1) *and* (N_2,A_2)*, it holds that*

$$\min_{k=1,2} \frac{C(k,n_k)}{1-P(k,n_k)} \le \frac{C(1,n_1)+P(1,n_1)C(2,n_2)}{1-P(1,n_1)P(2,n_2)} \le \max_{k=1,2} \frac{C(k,n_k)}{1-P(k,n_k)}.$$

Proof: We first examine the case where

$$\frac{C(1,n_1)}{1-P(1,n_1)} \le \frac{C(2,n_2)}{1-P(2,n_2)}. \tag{A1}$$

It follows that

$$C(1,n_1)(1-P(2,n_2)) \le C(2,n_2)(1-P(1,n_1))$$

$$C(1,n_1)P(1,n_1)(1-P(2,n_2)) \le C(2,n_2)P(1,n_1)(1-P(1,n_1))$$

and if we add $C(1,n_1)(1-P(1,n_1))$ to both sides of the inequality then

$$C(1,n_1)(1-P(1,n_1)P(2,n_2)) \le (C(1,n_1)+C(2,n_2)P(1,n_1))(1-P(1,n_1))$$

so that we obtain

$$\frac{C(1,n_1)}{1-P(1,n_1)} \le \frac{C(1,n_1)+P(1,n_1)C(2,n_2)}{1-P(1,n_1)P(2,n_2)}.$$

It can be shown by a similar argument that the second inequality of the lemma is also valid when equation (A1) holds. Analogously, the lemma can be shown to hold when the inequality in equation (A1) is reversed. □

***Proof of Theorem* 3**

Without loss of generality, we assume that Algorithm *PC* selects chain 1 first. Consider an optimal schedule **s** imposing a complete order on N (such **s** exists, by Theorem 1) that does not start with jobs [1,1] to $[1,\rho_1]$. This means that chain 1 is interrupted by activities from the other chains, and the corresponding order relation E can be represented as

$$< \beta_{1,1},\beta_{1,2},...,\beta_{1,m_1},\alpha_1,\beta_{2,1},\beta_{2,2},...,\beta_{2,m_2},\alpha_2,...,\beta_{L,1},\beta_{L,2},...,\beta_{L,m_L},\alpha_L,[1,\rho_1],\gamma >,$$

in which α_i are disjoint ordered subsets of consecutive jobs in chain 1 and β_{ji} are similar subsets each of a single chain different from chain 1; γ consists of the remaining activities of N. If we write $P(\alpha_i,|\alpha_i|)$ as $P(\alpha_i)$ and with similar conventions for argument β and function $C()$, application of the first part of Lemma A1 yields that

$$\frac{C(\beta_{1,1})}{1-P(\beta_{1,1})} \ge \frac{C(\beta_{1,2})}{1-P(\beta_{1,2})} \ge ... \ge \frac{C(\alpha_1)}{1-P(\alpha_1)} \ge \frac{C(\beta_{2,1})}{1-P(\beta_{2,1})} \ge ... \ge \frac{C(\beta_{L,m_L})}{1-P(\beta_{L,m_L})} \ge \frac{C(\alpha_L)}{1-P(\alpha_L)} \ge \frac{c([1,\rho_1])}{1-p([1,\rho_1])}. \tag{A2}$$

The foregoing set of inequalities is valid because no two consecutive subsets belong to the same chain, and so Lemma A1 can be applied to the two subsets in isolation. We therefore also have

$$\frac{C(\alpha_1)}{1-P(\alpha_1)} \ge \frac{C(\alpha_2)}{1-P(\alpha_2)} \ge ... \ge \frac{C(\alpha_L)}{1-P(\alpha_L)} \ge \frac{c([1,\rho_1])}{1-p([1,\rho_1])},$$

and repeated application of Lemma A2 then gives

$$\frac{C(\alpha_1)}{1-P(\alpha_1)} \ge \frac{C(1,\rho_1)}{1-P(1,\rho_1)},$$

and from the definition of ρ, equality must hold in the latter equation, and therefore also in the inequalities with arguments α_1 to $[1,\rho_1]$ in (A2). From the definition of ρ, we also see that the β_{1i} are not void only if

$$\frac{C(\beta_{1i})}{1-P(\beta_{1i})} = \frac{C(1,\rho_1)}{1-P(1,\rho_1)},$$

and therefore, according to the second part of Lemma A1, all the different components of chain 1 in schedule **s** up to $[1,\rho_1]$ can be moved to the front of the schedule without effect on the objective function. As a consequence, an optimal schedule exists that executes chain 1 contiguously up to job $[1,\rho_1]$ in first position. The set of activities $N \setminus \{0,n\}$ without the selected part of chain 1 can now be considered as a new smaller project, and the same reasoning can be followed iteratively to show that Algorithm *PC* produces an optimal schedule. The algorithm runs in time $O(n^2)$, because the computation of ρ_k for all chains k can be performed in $O(n)$ time, and this is repeated $O(n)$ times. □

In the proof of Theorem 4, we use the result of Lemma A3. Lenstra and Rinnooy Kan (1978), inspired by Lawler (1978), show that $1|prec|\Sigma_j w_j C_j$ is strongly *NP*-hard by means of a reduction from OPTIMAL LINEAR ARRANGEMENT (OLA), which is defined as follows (Garey and Johnson, 1979):

OPTIMAL LINEAR ARRANGEMENT (OLA)
Instance: Undirected graph $G(V,E)$ and positive integer K.
Question: Is there a one-to-one function $f: V \rightarrow \{1,2,...,|V|\}$ such that $\sum_{(u,v)\in E}|f(u)-f(v)| \leq K$?

OLA was shown to be *NP*-complete by Garey et al. (1976). Lenstra and Rinnooy Kan (1978) explain how to solve an arbitrary instance of OLA by means of a suitably constructed instance of $1|prec|\Sigma_j w_j C_j$. Their reduction fulfills the conditions for a polynomial transformation, such that $1|prec|\Sigma_j w_j C_j$ is *NP*-hard. We will construct a similar proof to establish *NP*-hardness of the *maximization* of the weighted sum of completion times $\Sigma_j w_j C_j$, with non-negative integer job durations d'_i and non-negative integer weights w_i for each $i \in J$, with J the set of jobs to be scheduled, C_i the completion time of job i, and precedence graph $G(J,F)$ with precedence constraints that are to be respected. We only allow semi-active solution schedules; remark that zero durations are allowed. The corresponding problem is referred to as Π.

Lemma A3. *Problem* Π *is NP-hard.*
Proof: For an input instance of OLA, we define the following instance of Π: $|J|=|V|+|E|$, and J contains 'vertex' jobs J_i, $i \in V$, with duration $d'_i=1$ and weight w_i equal to $|E|-u_i$, with u_i the degree of i in $G(V,E)$; and also 'edge' jobs $J_{\{i,j\}}$, $\{i,j\} \in E$, with $d'_{\{i,j\}}=0$ and $w_{\{i,j\}}=2$. In F are the precedence constraints $J_{\{i,j\}} \prec J_i$ for all $\{i,j\} \in E$. Consider any linear arrangement f of V, and suppose we schedule the corresponding vertex jobs in the Π-instance in increasing value of f. Given its zero duration, each

edge job $J_{\{i,j\}}$ can be assumed to immediately precede the earliest of its associated vertex jobs J_i and J_j without harmful effect on the objective function. The value of this schedule is therefore given by

$$\begin{aligned}\sum_{j\in J} w_j C_j &= \sum_{i\in V}(|E|-u_i)f(i) + \sum_{(i,j)\in E} 2\min\{f(i),f(j)\} \\ &= \sum_{i\in V}|E|f(i) + \sum_{(i,j)\in E}(2\min\{f(i),f(j)\} - f(i) - f(j)) \\ &= \left(\tfrac{1}{2}\right)|E|.|V|.(|V|+1) + \sum_{(i,j)\in E}(\min\{f(i),f(j)\} - \max\{f(i),f(j)\}) \\ &= \left(\tfrac{1}{2}\right)|E|.|V|.(|V|+1) - \sum_{(i,j)\in E}|f(i)-f(j)|.\end{aligned}$$

It follows that Π minimizes $\sum_{(i,j)\in E}|f(i)-f(j)|$, since the first term in the last expression is a constant, and so OLA has a yes-answer if and only if there is a schedule to Π with objective value $\geq \left(\tfrac{1}{2}\right)|E|.|V|.(|V|+1)-K$, which proves the decision problem version of Π to be *NP*-complete, and Π itself *NP*-hard. □

***Proof of Theorem* 4**

For an arbitrary instance of Π, we can construct an instance of *RDPSP*, as follows. The set of activities $N=J\cup\{n\}$, with $n=|J|+1$. We have negative activity cash flows $c_i=-w_i$, $\forall i\in J$, each duration $d_i=1$ and the set of precedence-related activity pairs $A=F$. $c_n=0$, $d_n=1$ and n is successor to all jobs without successor in $G(J,F)$. For each activity $i\in J$, we set probability $p_i = (1 - d'_i/M)$ with non-negative integer $M \geq d'_{max} = \max_{i\in J}\{d'_i\}$. Since $r=0$, an optimal solution to *RDPSP* that is not a complete order on N can be re-arranged in polynomial time into a complete order with equal objective function (as outlined in the proof of Theorem 1). Consider such an optimal complete order, and let $[i]$ represent the job in the i^{th} position. When $r=0$, the objective-function value of the thus built *RDPSP*-instance equals

$$c_{[1]} + \sum_{j=2}^{n-1} c_{[j]} \prod_{i=1}^{j-1} p_{[i]} = c_{[1]} + \sum_{j=2}^{n-1} c_{[j]} \prod_{i=1}^{j-1}\left(1-\frac{d'_{[i]}}{M}\right), \tag{A3}$$

with

$$\prod_{i=1}^{j-1}\left(1-\frac{d'_{[i]}}{M}\right) = 1 - \frac{1}{M}\sum_{k=1}^{j-1} d'_{[k]} + \frac{1}{M^2}\sum_{k=1}^{j-2}\sum_{l=k+1}^{j-1} d'_{[k]}d'_{[l]} - \ldots$$

so that the objective function (A3) becomes

$$\sum_{j=1}^{n-1} c_j - \frac{1}{M}\sum_{j=2}^{n-1} c_{[j]}\sum_{k=1}^{j-1} d'_{[k]} + \sum_{j=2}^{n-1} c_{[j]}\sum_{i=2}^{j-1}\left(\frac{-1}{M}\right)^i \sum_{\substack{\text{all sets } K:|K|=i,\\ K\subseteq\{1,2,\ldots,j-1\}}} \prod_{k\in K} d'_{[k]}.$$

The first term in this expression is a constant. We now require that the impact of a change of a single unit in quantity $\sum_{j=2}^{n-1} c_{[j]}\sum_{k=1}^{j-1} d'_{[k]}$ (the weighted sum of the starting times in Π) be larger than the largest possible change in all remaining terms, so that any optimal solution to the *RDPSP*-instance automatically optimizes this weighted sum. We need that

$$\frac{1}{M} > \sum_{j=2}^{n-1} c_{max}\sum_{i=2}^{j-1}\frac{1}{M^i}\sum_{\substack{\text{all sets } K:|K|=i,\\ K\subseteq\{1,2,\ldots,j-1\}}}\prod_{k\in K} d'_{max}, \tag{A4}$$

with $c_{max} = \max_{i \in J}\{|c_i|\}$. The right hand side of equation (A4) is smaller than or equal to

$$\sum_{j=2}^{n-1} c_{max} \sum_{i=2}^{j-1} \left(\frac{d'_{max}}{M}\right)^i \binom{j-1}{i}$$

and this expression in turn is strictly smaller than

$$c_{max}\left(\frac{d'_{max}}{M}\right)^2 2^{n-1}$$

since $d'_{max} \leq M$, $\sum_{i=2}^{j-1}\binom{j-1}{i} < 2^{j-1}$ and $\sum_{j=2}^{n-1} 2^{j-1} < 2^{n-1}$. This leads us to the conclusion that equation (A4) holds when

$$M = c_{max}\, d'^2_{max}\, 2^{n-1}.$$

For this M-value, we have shown that any job sequence maximizing equation (A3) also maximizes $\sum_{j=2}^{n-1} w_{[j]} \sum_{i=1}^{j-1} d'_{[i]}$, which equals the weighted sum of completion times minus constant $\sum_{i=1}^{n-1} w_i d'_i$. This description allows for a polynomial reduction from the *decision problem version* of Π to the *optimization problem RDPSP*. □

***Proof of Lemma* 1**

All intermediate activities are started as late as possible. The only reason why an activity would not end *exactly* at the start of its earliest starting successor in E_h, is because it needs to be in parallel with some other activity. If we iteratively remove all parallelity constraints for this activity and shift it later in time until it ends exactly at its earliest successor starting time, there is no effect on the contribution to the objective function of any of the other activities. On the other hand, the contribution of the activity itself to the objective function goes down, first of all because of the discounting effect, and second also because additional activities may now end before or at the starting time of the activity itself, thereby allowing to reduce its expected NPV via extra information flows. □

***Proof of Lemma* 2**

We describe a reduction from the strongly *NP*-hard problem MWIS (defined in Section 6.1) to the computation of $\mu_0^{(h)}$: we set $N=V$, $d_i=w_i$ ($i \in N$), $E_h=\varnothing$ and $\pi_h=C$. □

***Proof of Lemma* 3**

Each set V_i contains either zero or one vertex of V°, because V° is an independent set and V_i is a clique. Collect in set $B \subseteq \{1,...,t\}$ the indices of all the sets V_i that contain exactly one vertex of V°, and identify the corresponding vertices as v_i, $i \in B$. For each V_i, $i \in B$, $\max_{j \in V_i}\{w_i\} \geq w_{v_i}$, such that $\sum_{i=1,...,t} \max_{j \in V_i}\{w_j\} \geq \sum_{i \in B} \max_{j \in V_i}\{w_j\} \geq \sum_{j \in V^\circ} w_j$, and the lemma is seen to hold. □

References

Adlakha, V.G., V.G. Kulkarni. 1989. A classified bibliography of research on stochastic PERT networks: 1966-1987. *INFOR* **27**(3) 272-296.

Adler, P.S., A. Mandelbaum, V. Nguyen, E. Schwerer. 1995. From project to process management: an empirically-based framework for analyzing product development time. *Management Science* **41**(3) 458-484.

Adler, P.S., A. Mandelbaum, V. Nguyen, E. Schwerer. 1996. Getting the most out of your product development process. *Harvard Business Review* **March-April** 134-152.

Bard, J.F. 1985. Parallel funding of R&D tasks with probabilistic outcomes. *Management Science* **31**(7) 814-828.

Bartusch, M., R.H. Möhring, F.J. Radermacher. 1988. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research* **16** 201-240.

Busygin, S., D.V. Pasechnik. 2003. On $\chi(G) - \alpha(G) > 0$ gap recognition and $\alpha(G)$-upper bounds. *Electronic Colloquium on Computational Complexity (ECCC), Report TR03-052*. Available at ftp://ftp.eccc.uni-trier.de/pub/eccc/reports/2003/TR03-052/index.html.

Crama, P., B. De Reyck, Z. Degraeve, W. Chong. 2004. R&D project valuation and licensing negotiations at Phytopharm plc. *London Business School Working Paper*.

Demeulemeester, E., W. Herroelen. 2002. *Project Scheduling. A Research Handbook.* Kluwer Academic Publishers.

Demeulemeester, E., M. Vanhoucke, W. Herroelen. 2003. A random generator for activity-on-the-node networks. *Journal of Scheduling* **6** 13-34.

DiMasi, J.A., R.W. Hansen, H.G. Grabowski. 2003. The price of innovation: new estimates of drug development costs. *Journal of Health Economics* **2** 151-185.

Elmaghraby, S.E. 1977. *Activity Networks: Project Planning and Control by Network Models*. Wiley.

Eppinger, S.D., D.E. Whitney, R.P. Smith, D.A. Gebala. 1994. A model-based method for organizing tasks in product development. *Research in Engineering Design* **6** 1-13.

Garey, M.R., D.S. Johnson. 1979. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

Gassmann, O., G. Reepmeyer, M. von Zedtwitz. 2004. *Leading Pharmaceutical Innovation. Trends and Drivers for Growth in the Pharmaceutical Industry*. Springer-Verlag, Berlin Heidelberg New York.

Hagstrom, J.N. 1988. Computational complexity of PERT problems. *Networks* **18** 139-147.

Hill, A.V. 2002. The Encyclopedia of Operations Management Terms. *Available on-line as*: http://www.poms.org/POMSWebsite/EducationResources/omencyclopedia.pdf.

Hoedemaker, G.M., J.D. Blackburn, L.N. Van Wassenhove. 1999. Limits to concurrency. *Decision Sciences* **30**(1) 1-18.

Jørgenson, T. 1999. Project Scheduling - a Stochastic Dynamic Decision Problem. *Doctoral dissertation, Norwegian University of Science and Technology, Trondheim, Norway*.

Kavadias, S., C.H. Loch. 2004. *Project Selection under Uncertainty. Dynamically Allocating Resources to Maximize Value*. Kluwer Academic Publishers.

Krishnan, V., S. Bhattacharya. 2002. Technology selection and commitment in new product development: the role of uncertainty and design flexibility. *Management Science* **3** 313-327.

Krishnan, V., S.D. Eppinger, D.E. Whitney. 1997. A model-based framework to overlap product development activities. *Management Science* **43**(4) 437-451.

Lawler, E.L. 1976. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York.

Lawler, E.L. 1978. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics* **2** 75-90.

Lenstra, J.K., A.H.G. Rinnooy Kan. 1978. Complexity of scheduling under precedence constraints. *Operations Research* **26** 22-35.

Leus, R. 2003. The Generation of Stable Project Plans. Complexity and Exact Algorithms. *PhD Thesis, Katholieke Universiteit Leuven, Leuven, Belgium*.

Luh, P.B., F. Liu, B. Moser. 1999. Scheduling of design projects with uncertain number of iterations. *European Journal of Operational Research* **113** 575-592.

Lockett, A.G., A.E. Gear. 1973. Representation and analysis of multi-stage problems in R&D. *Management Science* **19**(8) 947-960.

Mastor, A.A. 1970. An experimental and comparative evaluation of production line balancing techniques. *Management Science* **16** 728-746.

Möhring, R.H. 2001. Scheduling under uncertainty: bounding the makespan distribution. *In*: Alt, H. (ed.). *Computational Discrete Mathematics, Advanced Lectures*. Lecture notes in Computer Science 2122, Springer.

Neumann, K., C. Schwindt, J. Zimmermann. 2003. *Project Scheduling with Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions*. Springer-Verlag, 2nd edition.

Parker, R., R. Rardin. 1988. *Discrete optimization*. Academic Press.

Pinedo, M. 2002. *Scheduling. Theory, Algorithms, and Systems*. Prentice-Hall, Inc., Upper Saddle River, NJ.

Stork, F. 2001. Stochastic Resource-Constrained Project Scheduling. *PhD Thesis, Technische Universität Berlin, Berlin, Germany*.

Zemel, D., I. David, A. Mehrez. 2001. On conducting simultaneous versus sequential engineering activities in risky R&D. *International Transactions in Operational Research* **8** 585-601.